REPORT

Computer Science Technical Report: **95-20** **February 1995**

Secure Communication in FRTP

Tage Stabell-Kulø

INSTITUTE OF MATHEMATICAL AND PHYSICAL SCIENCES

Department of Computer Science

University of Tromsø, N-9037 TROMSØ, Norway, Telephone +47 77 64 40 41, Telefax +47 77 64 45 80

Abstract

To ensure privacy, users of the File Repository Transfer Protocol (Frtp) may require that communication partners are properly authenticated. If one partner wants the communication to be crypted, the other must honor the request. This report describes how authentication and privacy are realized. It is shown, by means of an authentication logic, that the protocol achieves its goal in this respect.

Contents

1 Introduction

The *File Repository* (Fr) is part of the Pasta project.[1] The aim of Pasta is to gain better insight into the fundamental problem of partitioned data. The vehicle is the Fr utilized by portable computers. With Fr we will investigate disconnected operations via pessimistic concurrency control.

Servers running the Fr provide users access to storage through the File Repository Transfer Protocol (Frtp). For simplicity, Frtp is Ascii-based, that is, all commands and responses are in Ascii. This makes it less hard to debug clients and servers. On the other hand it makes equally easy to eavesdrop on the communication. Any simple network-sniffer will do, since all commands and responses are sent without any encoding (in Ascii). In the developing and "debugging" phase this is not a problem, but as the repository is used on a regular basis, users wants privacy. Even if there is a desire for privacy in general, transporting files across networks belonging to other organizations places focus on the communication as such. This paper describes the functionality included in Frtp to ensure secure communication.

This report starts out by describing the relevant parts of Frtp, and show some examples of how the protocol is used. In section 2, we sketch how authentication is performed and privacy obtained. The emphasis is is placed on describing the protocol and we give only informal description on the issues concerning authentication and privacy. In section 3 we give a short overview of the logic we use, and in section 4 we describe how the public keys of users and servers are distributed. The goals of the protocol (of interest to us here) are presented in section 5. Section 6 contains the proofs that the requirements set forth in section 5 are indeed met. In other words, we show that the protocol really authenticates the two parties, and it does so without being prone for replay or other attacks.

This report does not contain any new results, its aim is simply to give the reader confidence in the secrecy provided by Frtp, i.e. the secrecy is as secure as the keys and the cryptographic algorithms used.

We do not include any implementation details as the emphasis in this report is on the Helo and Secure protocols per se. This work is build upon Frtp version 1, and is planned to be included in Frtp version 2. Version 2 will be downwards compatible with version 1.

2 File Repository Transfer Protocol

The Fr is a software system that is design to support users of portable computers. It does so by providing a repository in which users can store files. The system enforces currency-control and manages state necessary to facilitate cooperation between users (of files) even without being able to use "call back" to ensure consistency. The users interact with the system through Frtp. The Fr is realized as servers running at the sites providing this service to users.

The system is based on the notion of servers and clients. Servers store files, and information about file, on behalf of clients. This results in a clear command–response style of communication. Servers do not initiate any activity. The overall protocol is modelled after Nntp and Smtp in this respect.

[1] Pasta is a joint effort between researchers at the Dipartimento di Ingegneria dell'Informazione, Università di Pisa, Italy and Seksjon for Informatikk, Universitetet i Tromsø, Norway.

```
S: (listens at a reserved TCP port)
C: (requests connection on this reserved TCP port)
S: 7352564846:423 Welcome to server Barf
C: CREATE foo
S: 230 File Created
C: LIST
S: 240 Directory read - data follows
S: foo
S: bar
S: FooBar
S: .
C: Delete foo
S: 231 File deleted
```

Figure 1: *Example of* FRTP*. In the example, "*`S:`*" indicates statements made by the server and "*`C:`*" indicate statements made by the client. Notice how a single "." (dot) terminates a series of lines of data sent from the server to the client. This is inherited from* SMTP *and* NNTP.

A sequence of commands and responses in which a client creates a file, lists a directory and deletes the same file again, is shown in Figure1.

Since files can contain arbitrary bit patterns the actual file data is sent between the server and the client outside the command stream. There are 14 commands in total. Only two of them are of interest to us here: HELO that starts the sequence of messages leading to mutual authentication, and SECURE that initiate private (encrypted) communication.

2.1 Helo

Authentication is not, per se, required by FRTP. But it is required that the server honor a request from a user that wants to authenticate it (and thereby authenticating itself) and a user may find that a server has been configured in such a way that it will only honor requests from users it can authenticate.

The user signals to the server desire for mutual authentication by means of the HELO command. The usage of HELO is shown in Figure 2. The first encrypted message (from the server) contains the user's nounce (the string "UserNounce") encrypted with the servers private key, followed by a blank line and then a new nounce ("ServerNounce"). The client answers by sending the servers nounce back, encrypted with it's own private key.

Informally, the server has proven who it is by making a signature (on the client's nounce) and the client does the same thing with the server's nounce. Observe that this protocol only ensures authentication, and that secure communication is assumed.

2.2 Secure

If the client, or the server, feels the need for privacy, it can ask the other party for it by means of the SECURE command. The command need not be honored, the necessary resources may not be available, in which case the connection can be terminated; an example of its usage can be found in Figure 3. Informally, the two parties prove who they are by signing each others

```
C: HELO Tage_Stabell-Kulø UserNounce
S: 350 Please respond to challenge
S: hIwCFlYLfgsv0BytouKCwG071vWPsj3QzkCjIQiMylX7YmvVTrPZV0b6
S: LueJcQsiaqy9Nzh6hWRqT6yphJ1mf1EpzATWEBBb94s2Qt2dDy6VGWHH
S: 7fnWISRbxDPJDkN76/sjTFkMaas3UZgM7cHd2yVxNsHeSP7Hisc=
S: =CDd6
S:
S: ServerNounce
S: .
C: HeloResponse
S: 351 Please send response, end with <CR-LF>.<CR-LF>
C. fACJz9fNy6N7chqxfNB8Nd6nEEwV9I1e56i1nQ26XUdetsBMd9KPbr3K
C: XueJcQsiaqy9Nzh6hWRqTeyphJ1mf1EpzATWEBBb94s2Qt2dDy6VGhdK
C: BKptUFtxXpL17G7IPiIIBBXicDGMu6uvUNCg1kHf0V+BdEasdGs=
C: =Gdaf
C: .
S: 250 Authentication OK
```

Figure 2: *The* HELO *command. The two large messages are "armored" encoding of a bit-stream, converting it to a format suitable for usage in an* ASCII *protocol.*

nounces. The user's nounce will be used to encrypt all subsequent communication (session key). The FRTP protocol will be visible only after decryption. Note that the "." (dot), a required part of text-messages in FRTP, is sent in clear text. This is to make it possible for the parties to know when a message has ended, and to make crypto-analysis harder. The SECURE protocol is modelled after [Denning81, p. 535].

3 Logic of Authentication

FRTP will be used in two settings with different assumptions:

- Where the two parties can assume that the network provides privacy. With this support of this kind, only authentication is needed;
- Where no assumptions are made. Both authentication and exchange of a session key are necessary.

The former is typically used when two machines are connected by modems that provides hardware encryption—such modems are commonplace—or where the operating system ensures the privacy of communication. As authentication on a secure channel is trivial—the assumptions one makes are powerful—the proof is short. In the latter case, the proof is still quite short since we have chosen a solution to the problem related to distribution of encryption keys that gives us an advantageous staring positions.

We use the logic presented in [Burrows90], with the notation used by the same authors in [Burrows94]. We will summarize the postulates and rules we will need in our analysis of the two protocols. The original work, that is [Burrows90], contains more rules and postulates than

```
S: 74623629:253 Welcome to server Foo
C: CREATE bar
S: 321 I only speak to users I know
C: SECURE
S: 260 Please send challange, end with <CR-LF>.<CR-LF>
C. CcJz9fNy6N7chqxfNB8Ni6nEEwV9I1e56i1nQ26XU/etsBMd9KPbr3K
C: LueJcQsiaqy9Nzh6hWRqT6yphJ1mf1EpzATWEBBb94s2Qt2dDy6VGWHH
C: 7fnWISRbxDPJDkN76/sjTFkMaas3UZgM7cHd2yVxNsHeSP7Hisc=
C: =CDd6
C: .
S: eKE1jmOCVWExLNAAh3GkPe+nJU+IhkAK6FXKJobWfhARBN6xWCQCXaYA
S: AACJz9fNy6N7chqxfNB8Ni6nEEwV9I1e56i1nQ26XU/etsBMd9KPbr3K
S: LueJcQsiaqy9Nzh6hWRqT6yphJ1mf1EpzATWEBBb94s2Qt2d
S: =Lfsv
S: .
```

Figure 3: *The* SECURE *command. As with the* HELO *command, the two large messages are "armored" encoding of a bit-stream, converting it to a format suitable for usage in an* ASCII *protocol.*

we present here. But we add one new postulate and include in the discussion the rationale behind it.

- The *message-meaning* rules concerns the interpretation of messages. For shared keys, we believe that if P and Q share a secret key K, and P sees a message encrypted by K, P believes that Q encrypted it.

  $$\frac{P \text{ b}\mathit{elieves}\ Q \overset{K}{\leftrightarrow} P, P \textbf{ sees } \{X\}_K}{P \textbf{ believes } Q \textbf{ said } X} \qquad (P1)$$

  For a public-secret key-pair we assume that data encrypted with one part of the key can be decrypted with the other. We use K^{-1} to denote the secret part and K for the public part of a key-pair (K, K^{-1}). If X is encrypted with the public part, we write $\{X\}_K$, and if X encrypted with the secret part we write $\{X\}_{K^{-1}}$.

  We have two postulates on key-pairs. The first states that the owner of the secret key can decrypt data encrypted with the public part.

  $$\frac{P \textbf{ believes} \overset{K}{\mapsto} P, P \textbf{ sees } \{X\}_K}{P \textbf{ sees } X} \qquad (P2)$$

  If data is encrypted with the secret part, that is, it can be decrypted with the public part, we assume the owner of the secret key did the encryption. We refer to the encryption with the secret part of a key-pair as a signature.

  $$\frac{P \textbf{ believes } \overset{K}{\mapsto} Q, P \textbf{ sees } \{X\}_{K^{-1}}}{P \textbf{ believes } Q \textbf{ said } X} \qquad (P3)$$

- For *nounce verification*, if P believes that if a message X is *fresh* and that Q said X, P believes that Q believes X. *Fresh* indicates that the value has not been used before, and that it is a random value.

$$\frac{P \textbf{ believes fresh}(X), P \textbf{ believes } Q \textbf{ said } X}{P \textbf{ believes } Q \textbf{ believes } X} \tag{$P4$}$$

  P **believes** Q **believes** X means that P believes that Q is taking part in *this* run of the protocol—it is not a replay of an earlier run.

- We assume that a message, containing two (or more) parts, and that has been encrypted (or signed), is treated as an integral message—it is not possible to merge two encrypted messages into a single new one, or, split an encrypted message into smaller parts without knowing the relevant keys.

$$\frac{P \textbf{ believes } Q \textbf{ believes}(X, Y)}{P \textbf{ believes } Q \textbf{ believes } X} \tag{$P5$}$$

  The same holds for freshness:

$$\frac{P \textbf{ believes fresh}(X)}{P \textbf{ believes fresh } (X, Y)} \tag{$P6$}$$

  Notice that if P **believes fresh**(X) and P **sees**(X, Y) it is not valid to conclude **fresh**(Y).

- On jurisdiction we postulate that if P believes that Q controls X and P believes Q believes X, P believes X as well:

$$\frac{P \textbf{ believes } Q \textbf{ controls } X, P \textbf{ believes } Q \textbf{ believes } X}{P \textbf{ believes } X} \tag{$P7$}$$

  For nounces this implies that P have faith in Q's ability and willingness to generate proper values.

- If P believes that a key K is shared with Q, and sees X encrypted with K, P will believe that Q did the encryption (Q *once* said X). This is the postulate on message meaning ($P1$). If K is fresh, Q must have done the encryption recently, at least after K was generated; the fact that the encryption took place ensures P that Q knows the key K.

$$\frac{P \textbf{ believes } Q \overset{K}{\leftrightarrow} P, P \textbf{ believes fresh}(K), P \textbf{ sees } \{X\}_K}{P \textbf{ believes } Q \textbf{ believes } Q \overset{K}{\leftrightarrow} P} \tag{$P8$}$$

  As a corollary we note that if P believes the key K is fresh and shared with Q, and sees X encrypted with K, P will believe that Q encrypted X now, that is, Q believes X.

$$\frac{P \textbf{ believes } Q \overset{K}{\leftrightarrow} P, P \textbf{ believes fresh}(K), P \textbf{ sees } \{X\}_K}{P \textbf{ believes } Q \textbf{ believes } X} \tag{$P8'$}$$

  In [Burrows90, p. 7] it is said that nounce-verification ($P4$) is the only way to promote P **believes** Q **said** X into P **believes** Q **believes** X, and the freshness of X is used.

In ($P8'$) we use a fresh key instead. However, notice that P can only verify that X is encrypted with K when X already is known to P.

The two postulates ($P8$) and its corollary ($P8'$) are new and do not appear in [Burrows90], but they can be viewed as just a combination of the postulate on message-meaning ($P1$) and on nounce-verification ($P4$).

In the following we will refer to these postulates as we use them.

4 Key Distribution

The Fr relies on both public-key and shared-key cryptography in order to achieve its goals. The protocol is designed to meet the needs in a well defined environment, and key-distribution reflects this.

Most authentication protocols rely on a trusted third party. This trust must be general: the party is trusted to keep its private key secret, generate "good" session keys upon request, not share with others their nounces and keys meant to be kept secret and to execute the protocol faithfully.

In general, communication with a third party is not possible in for users of the Fr, as the system is designed to also support portable computers. That is: portables will often connect to the servers using modems. In this situation it is impossible to contact a third party without in-curing high costs, particularly in time. In other words, precautions must be taken to ensure that the user either have the server's key available, or have the possibility to infer trust in the server's key. This must also be the case when the credentials are presented by (someone who claims to be) the server itself.

When a person wants to become a user of the Fr, she will have to contact an administrator. Regarding the servers public key, the users has two (not mutual exclusive) options. First, she can copy it. This way she can later readily verify the servers signatures. If this is undesirable (possibly due to constraints on disk space) she can also choose to sign the key. The latter is a proof that she trusts the ownership of the key.

In the former case, the user U will believe that the K_s indeed is the public key of S. The belief is established during a personal meeting.[2]

In the latter case, the user gave the server the certificate $\{S, K_s\}_{K_u^{-1}}$. When it later is presented to the user, presumably by the server, the user (and anyone with access to the user's public key) can verify that U believed that K_s was S's public key. The string "S" in the certificate should probably include a time-stamp to ensure that certificates containing old, possibly compromised keys, can be recognized.

This technique for the user to obtain the server's key is not (yet) part of Frtp, and it will not be given further treatment; we will in the following assume that U **believes** $\overset{K_s}{\mapsto} S$. Notice that in cases where the user has good connectivity, the key-distribution can be done by a variety of protocols [Liebl93].

[2]Again, it must be stressed that both principals are assumed to be honest.

5 The Goals

The exchange of messages has as its goal in get to a situation where the server and the client have trust in each others presence. The authentication-phase is complete when

$$U \textbf{ believes } S \textbf{ believes } X \tag{$HG1$}$$

and

$$S \textbf{ believes } U \textbf{ believes } X. \tag{$HG2$}$$

This means that both the server and the user believes that the other party has seen the message X, that is, they both believe that the other party exists (took part in the authentication.) This is of no use unless the partners trust the channel that connects them. For privacy, in addition, one must exchange a session-key for encryption.

$$U \textbf{ believes } S \overset{K}{\leftrightarrow} U \tag{$SG1$}$$

$$S \textbf{ believes } U \textbf{ believes } S \overset{K}{\leftrightarrow} U \tag{$SG2$}$$

$$S \textbf{ believes } S \overset{K}{\leftrightarrow} U \tag{$SG3$}$$

$$U \textbf{ believes } S \textbf{ believes } S \overset{K}{\leftrightarrow} U \tag{$SG4$}$$

That is, a shared key (K_{su}) must be known to both parties, and they must both believe that the other party also knows the key.

We will show that the HELO command and its responses, leads the server and the user to a situation where ($HG1$) and ($HG2$) holds. Furthermore, we will show that after having gone through the SECURE command and its responses, ($SG1$–$SG4$) all holds.

6 Analysis

We will proceed in our analysis in the same way as in [Burrows90]. The analysis consists of three parts: describe the protocol, idealize it, and the analysis proper. As is usual, we assume that both parties are able, and willing, to execute the protocol faithfully; threats against secrecy originates from a hostile third party.

6.1 The Helo protocol

Section 2.1 we showed the protocol as it will look to an implementor. The essence is:

Message 1	$U \rightarrow S$	:	U, N_u
Message 2	$S \rightarrow U$	:	$\{N_u\}_{K_s^{-1}}, N_s$
Message 3	$U \rightarrow S$	:	$\{N_s\}_{K_u^{-1}}$

The HELO protocol will only be used on communication-channels that already provide privacy. This means that no-one else can intrude on the line without being detected, for example, by the fact that an intruder can only disturb the activity. Since the line is assumed private, an intruder can not insert any valid data or commands. This also has the implication that non-encrypted parts of the protocol need not be removed in the idealized protocol. The idealized protocol is as follows:

Message 2	$S \rightarrow U$	:	$\{N_u\}_{K_s^{-1}}, N_s$
Message 3	$U \rightarrow S$	:	$\{N_s\}_{K_u^{-1}}$

Note that the two parts of Message 2 belongs together as if the message had been $\{N_U, N_S\}_{K_S^{-1}}$, in that they are linked together even without having to be encrypted together. In general encryption is computationally expensive and should not be done when avoidable.

We have the following assumptions:

$$U \textbf{ believes } \overset{K_s}{\mapsto} S$$

$$S \textbf{ believes } \overset{K_u}{\mapsto} U$$

$$U \textbf{ believes fresh } N_u$$

$$S \textbf{ believes fresh } N_s$$

Message 1 does not lead any of them to believe anything. By using the *message-meaning* postulate $(P1)$, Message 2 gives

$$\frac{U \textbf{ believes } \overset{K_s}{\mapsto} S, U \textbf{ sees } \{N_u\}_{K_s^{-1}}}{U \textbf{ believes } S \textbf{ said } N_u}$$

and by using the *nounce-verification* rule $(P4)$ we obtain

$$\frac{U \textbf{ believes fresh } N_u, U \textbf{ believes } S \textbf{ said } N_u}{U \textbf{ believes } S \textbf{ believes } N_u} \qquad (HG1)$$

which is what we desire for U. Symmetrically, by using $(P1)$, Message 3 gives

$$\frac{S \textbf{ believes } \overset{K_u}{\mapsto} U, S \textbf{ sees } \{N_s\}_{K_u^{-1}}}{S \textbf{ believes } U \textbf{ said } N_s}$$

and by again using the *nounce-verification* rule $(P4)$ we obtain

$$\frac{S \textbf{ believes fresh } N_s, S \textbf{ believes } U \textbf{ said } N_s}{S \textbf{ believes } U \textbf{ believes } N_s} \qquad (HG2)$$

which is what we desire for S. Both S and U now knows that the other party is present *now*.

6.2 The Secure protocol

Informally, the SECURE protocol is used between a server and a user when they, through an insecure network, want to establish a secure and authenticated channel. This channel will be a shared secret: a session key which will be used as a shared key in symmetric encryption. As with the HELO protocol, the key-distribution problem is assumed to be solved as described in section 4. The protocol will achieve making both parties believe that the other party is taking part in the protocol (authentication), and exchange a session key that will be used throughout the duration of the session to encrypt communication (and thereby creating the secure channel between them).

In contrast to most other settings, the two parties can-not contact a mutual trusted third party in order to facilitate authentication. This also imply that the session key must be generated by one of the parties; the user will generate the key. This means that

$$S \textbf{ believes } U \textbf{ controls } U \overset{K}{\leftrightarrow} S$$

In [Burrows90, p. 26] it is noted that, in the general case, this is a dubious assumption; the question is not whether U is *willing* to generate a good key, but if it is *competent*. In our setting, however, any data transmitted over the (secure) channel will belong to, or being controlled by the user. If the user chooses a bad session key, it will not represent a threat to other users data or the server's integrity. For this reason we let the user, as opposed to the server, choose the key.

The SECURE protocol can be described as follows:

Message 1:	$S \to U$	:	T
Message 2:	$U \to S$	:	$U, \{\{S, T, K_{su}\}_{K_u^{-1}}\}_{K_s}$
Message 3:	$S \to U$	:	$\{T\}_{K_{su}}$

The first message appears *before* the SECURE command, as one can see in the example 3 (on page 4). It is the time-stamp given by the server as part of the initial "Welcome" message, but logically it is part of the secure protocol. It is the current time followed by a serial number.

In order to give the formalized protocol, we first state the assumptions:

$$S \textbf{ believes } \overset{K_s}{\mapsto} S$$

$$S \textbf{ believes } \overset{K_u}{\mapsto} U$$

$$S \textbf{ believes } U \textbf{ controls } S \overset{K}{\leftrightarrow} U$$

$$U \textbf{ believes } S \textbf{ controls } T$$

$$U \textbf{ believes fresh}(K_{su})$$

When S believes that a time-stamp T is fresh, it implies that the granularity of the clock by which it is generated must be finer than the rate by which connections are established; S must ensure that all T indeed are fresh (not used before). In [Abadi94, section 6], principle 7 reads

> The use of a predictable quantity (such as the value of a counter) can serve in guaranteeing newness, through a challenge-response exchange. But if a predictable quantity is to be effective, it should be protected so that an intruder cannot simulate a challenge and later replay a response.

In the SECURE protocol, if the value T was used as a challenge, an intruder can deceive S by acting as follows. When U tries to contact S the intruder intercepts the call answers with a future value of T. The intruder records U's reply. Later, at the correct moment, the intruder connects to S and, after having seen the previously used value of T, it sends the reply it previously got from U. S will now, incorrectly, believe that *it* communicated with U. Even though the intruder can not obtain any service from S (in the role of U), the false information on the presence of U can lead S into disarray.

To render this impossible, the time is concatenated with a nounce; the colon separates the two. We assume:

$$S \textbf{ believes fresh}(T)$$

In addition we will assume that initially one of the goals holds

$$U \textbf{ believes } S \overset{K}{\leftrightarrow} U \tag{SG1}$$

Notice that the key to be used as a session key between S and K (the key K_{su}) is written as $S\overset{K}{\leftrightarrow}U$ in the idealized protocol when it represent the existence of the key, but as K_{su} when it is used.

The idealized protocol then becomes:

Message 2: $U \rightarrow S \quad : \quad \{\{T, S\overset{K}{\leftrightarrow}U\}_{K_u^{-1}}\}_{K_s}$

Message 3: $S \rightarrow U \quad : \quad \{T\}_{K_{su}}$

The name of S has been removed from message 2 as it does not change the proof. The rationale for its presence in the protocol is discussed below.

With the postulates for public keys ($P2$), Message 2 gives

$$\frac{S \textbf{ believes } \overset{K}{\mapsto}S, S \textbf{ sees } \{\{T, S\overset{K}{\leftrightarrow}U\}_{K_u^{-1}}\}_{K_s}}{S \textbf{ sees } \{T, S\overset{K}{\leftrightarrow}U\}_{K_u^{-1}}}$$

With the *message-meaning* postulate for public keys ($P3$) we obtain

$$\frac{S \textbf{ believes } \overset{K_u}{\mapsto}U, S \textbf{ sees } \{T, S\overset{K}{\leftrightarrow}U\}_{K_u^{-1}}}{S \textbf{ believes } U \textbf{ said } (T, S\overset{K}{\leftrightarrow}U)}$$

Since it is assumed that S **believes fresh**(T), the postulate on *combined freshness* ($P6$) we obtain

$$\frac{S \textbf{ believes fresh}(T)}{S \textbf{ believes fresh}(T, S\overset{K}{\leftrightarrow}U)}$$

The postulate on *nounce verification* ($P4$) then lead to

$$\frac{S \textbf{ believes } U \textbf{ said } (T, S\overset{K}{\leftrightarrow}U), S \textbf{ believes fresh}(T, S\overset{K}{\leftrightarrow}U)}{S \textbf{ believes } U \textbf{ believes } (T, S\overset{K}{\leftrightarrow}U)}$$

with the postulate for *combined blief* ($P5$):

$$\frac{S \textbf{ believes } U \textbf{ believes}(T, S\overset{K}{\leftrightarrow}U)}{S \textbf{ believes } U \textbf{ believes}(S\overset{K}{\leftrightarrow}U)} \qquad (SG2)$$

If Message 2 had been

Message 2: $U \rightarrow S \quad : \quad T, \{S\overset{K}{\leftrightarrow}U\}_{K_u^{-1}}$

(T not signed together with $S\overset{K}{\leftrightarrow}U$) we could have derived S **believes** U **said** $S\overset{K}{\leftrightarrow}U$ but not S **believes fresh** $S\overset{K}{\leftrightarrow}U$. This conveys the situation where an intruder intercepts the message, removes U's good key and insert an old, presumably compromised, key.

Furthermore, if Message 2 had been

Message 2: $U \rightarrow S \quad : \quad \{\{T, K_{su}\}_{K_u^{-1}}\}_{K_s}$

(the name of S not included in the signed part of the message) a server S can deceive the user U by connecting to another server S' and forward to U the time-stamp presented to S by S'. When U sends the session-key to S, S simply remove the encryption, re-encrypt with the public key of S' and forward the key. S' will now believe that it talks to U. Including the name of the server is in accordance with [Abadi94, section 4], where principle 3 reads

If the identity of a principal is essential to the meaning of a message, it is prudent to mention the principal's name explicitly in the message.

Although we did assume that the participants are honest and strive to execute the protocols faithfully, having the name of the intended server in the certificate makes it possible to store data also on untrusted servers since the server can-not do more wrong than destroy or give away the data; an untrusted server is not able to modify or delete data stored on more trusted servers.

Message 2 contains U, but this has been removed from the idealized protocol since it is not bound to anything by encryption. It's presence makes it possible for S to know which user's public key to use.

If we continue, the rule for *juristiction* ($P7$) gives

$$\frac{S \textbf{ believes } U \textbf{ controls } S\overset{K}{\leftrightarrow}U,\ S \textbf{ believe } U \textbf{ believe } S\overset{K}{\leftrightarrow}U}{S \textbf{ believe } S\overset{K}{\leftrightarrow}U} \qquad (SG3)$$

When U receives Message 3, by using the postulate on fresh keys ($P8$), we get:

$$\frac{U \textbf{ believes } S \overset{K}{\leftrightarrow} U, U \textbf{ believes } \textbf{fresh}(K_{su}), U \textbf{ sees } \{T\}_{K_{su}}}{U \textbf{ believes } S \textbf{ believes } S \overset{K}{\leftrightarrow} U} \qquad (SG4)$$

In the FRTP protocol, T shall be the time, as known to S, and if we use the corollary of $P8$, the rule for fresh keys, $P8'$ we obtain

$$\frac{U \textbf{ believes } S \overset{K}{\leftrightarrow} U, U \textbf{ believes } \textbf{fresh}(K_{su}), U \textbf{ sees } \{T\}_{K_{su}}}{U \textbf{ believes } S \textbf{ believes } T}$$

and with the rule for jurisdiction ($P7$) we get

$$\frac{U \textbf{ believes } S \textbf{ controls } T, U \textbf{ believes } S \textbf{ believes } T}{U \textbf{ believes } T}$$

This implies that U can perform a loose synchronize its clock with S by subtracting the time spend executing the protocol.

7 Acknowledgements

Arne Helme provided valuable input in the initial phase of protocol extension described in this paper. Sape Mullender came up with the idea that made us use a fresh key to promote **said** into **believes**. The members of the PASTA project, both in Pisa and Tromsø, have provided excellent comments in their numerous readings of drafts.

Support has been received from "The Radical Initiative" of CaberNet, Esprit BRA 6361; it is gratefully acknowledged.

References

[Abadi94] Martin Abadi and Roger Needham. Prudent Engineering Practice for Cryptographic Protocols. Technical report 125. Digital Equipment Corporation Systems Research Center, Palo Alto, CA, June 1994. a preliminary version of this paper has appeared in the Proceedings of the 1994 IEEE Computer Society Symposium on Research in Security and Privacy.

[Burrows90] M. Burrows, M. Abadi, and R. Needham. A Logic of Authentication. *ACM Transactions on Computer Systems*, **8**(1):18–36, February 1990. Also available as DEC SRC Research Report 39, originally published February 28, 1989, and revised on February 22, 1990.

[Burrows94] Michael Burrows, Martin Abadi, and Roger Needham. The scope of a logic of authentication. Technical report 39 (Appendix). Systems Research Center, Digital Equipment Corporation, 13 May 1994.

[Denning81] Dorothy E. Denning and Giovanni Maria Sacco. Timestamps in key distribution protocols. *Communications of the ACM*, August 1981.

[Liebl93] Armin Liebl. Authentication in distributed systems: A bibliography. *Operating Systems Review*, **27**(4):31–41. ACM, October 1993.